The Behavior, Constraint, and Scenario (BeCoS) Tool:

A Web-Based Software Application for Modeling Behaviors and Scenarios

Justin Kaderka, Matthew Rozek, John Arballo, David Wagner, and Michel Ingham

Jet Propulsion Laboratory, California Institute of Technology

AIAA InfoTech@Aerospace, January 8-12, 2018

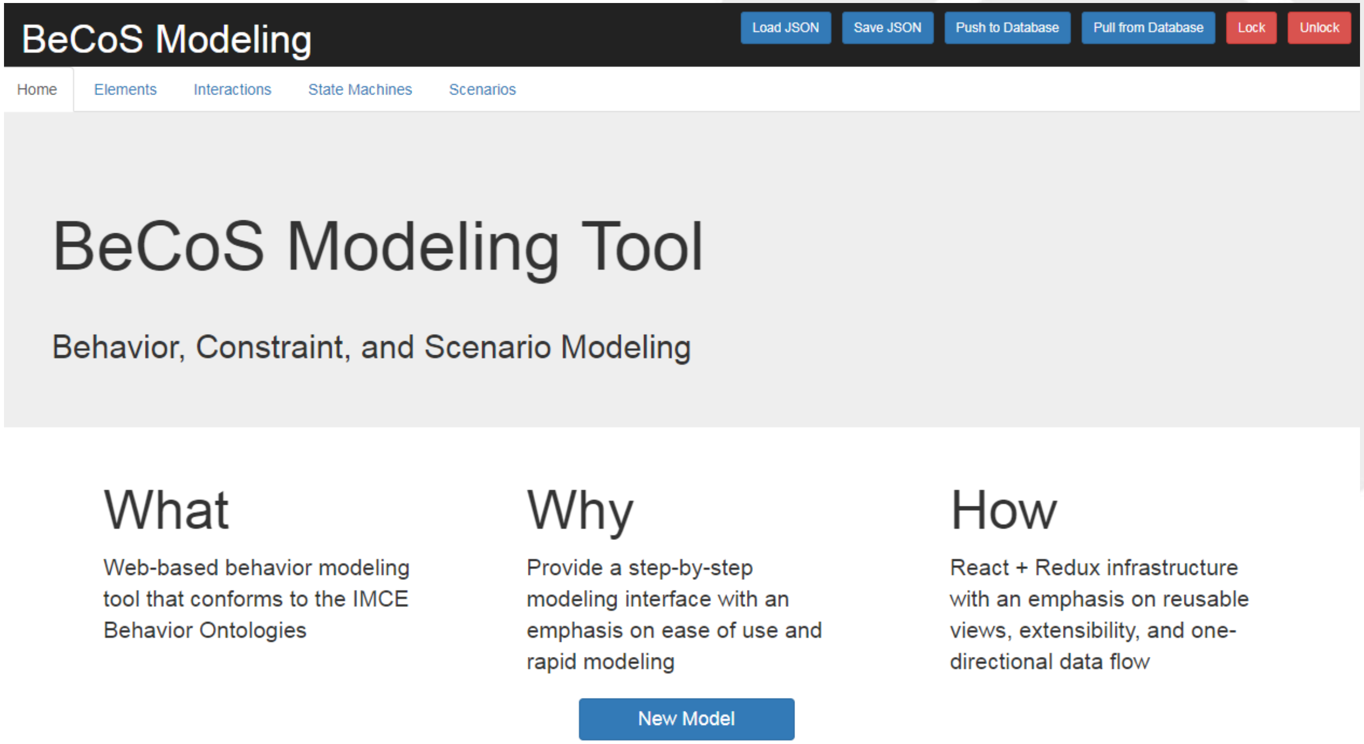
Kissimmee, FL

Outline

- Motivation and Background
- Ontology Definitions
- BeCoS Tool Overview
- Future Work
- Integrating BeCoS with JPL Tools
- Conclusions

BeCoS Modeling Tool

- Web-based application to model (i.e. specify) system **behavior** and **scenarios**



BeCoS Modeling Tool

Behavior, Constraint, and Scenario Modeling

What

Web-based behavior modeling tool that conforms to the IMCE Behavior Ontologies

Why

Provide a step-by-step modeling interface with an emphasis on ease of use and rapid modeling

How

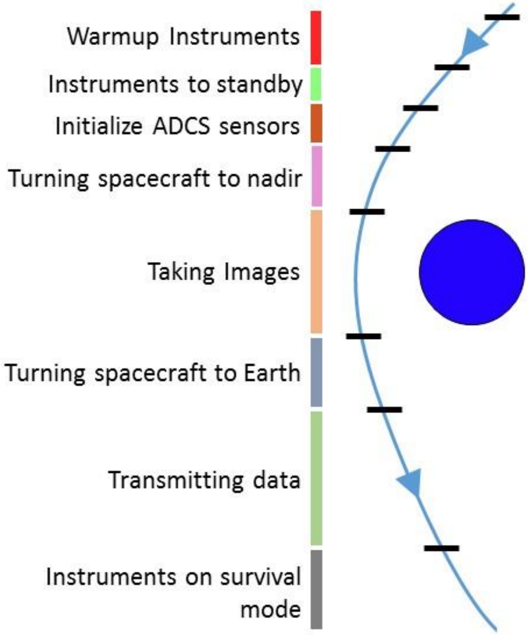
React + Redux infrastructure with an emphasis on reusable views, extensibility, and one-directional data flow

Motivation – What is meant by “Behavior”?

- State variable properties of a system and their values over time, and constraints that describe or define changes

- Classified into:
  - Intrinsic behavior - What variables are used to describe a system
  - Scenario behavior - How state variables evolve over time

[Scenario ICs: All OFF except Flight Computer and Power Regulator]



Spacecraft Powered Components (n=14)
Thruster Heaters
Camera Heater
Infrared Spectrometer Heater
Radio Transmitter
Downlink Amplifier
Power Bus Regulation / Conditioning Losses
Flight Computer
Star Tracker
Inertial Measurement Unit
Thruster Bank 1
Thruster Bank 2
Thruster Bank 3
Camera / Visual Imager
Infrared Spectrometer

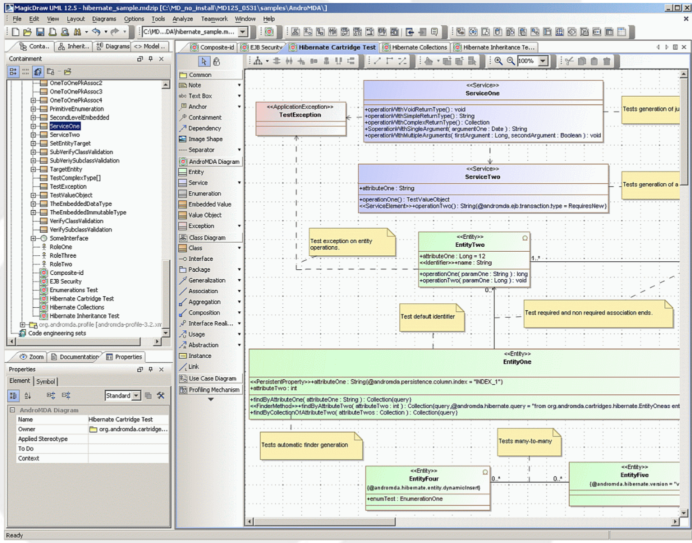
Motivation – Why Model Behavior?

- Need to understand how component / subsystem behavior aggregates into behavior of a real system

- Can run simulations on modeled behavior
  - Developmental phase
    - Power/Data predictions; assess against allocations
    - Science collections; assess against objectives
  - Operational phase
    - Simulate next operational cycle to ensure activities within resource constraints

Motivation – Issues with existing tools

- Behavior modeling in SysML using MagicDraw
  - MagicDraw is the JPL-supported tool for SysML
  - Relative high barrier to entry with SysML / MagicDraw for non-experienced users
    - Heavy-weight tool that presents to users much more expressivity (thus options) than needed for behavior modeling
  - There are existing domain-specific ontologies (e.g. Behavior Ontology)
    - There exists a much more compact representation in a modeling tool that is based on those ontologies
  - No general SysML representation for Temporal Constraint Networks (scenarios); must embed TCNs within activity diagram
- Scenario specification – declarative vs. imperative
  - Declarative – goal-based (what you want to happen)
  - Imperative – procedure-based (how you want something to happen)
  - Many current tools support imperative specification
    - Cumbersome to review with complex systems
    - Operational intent has to be inferred

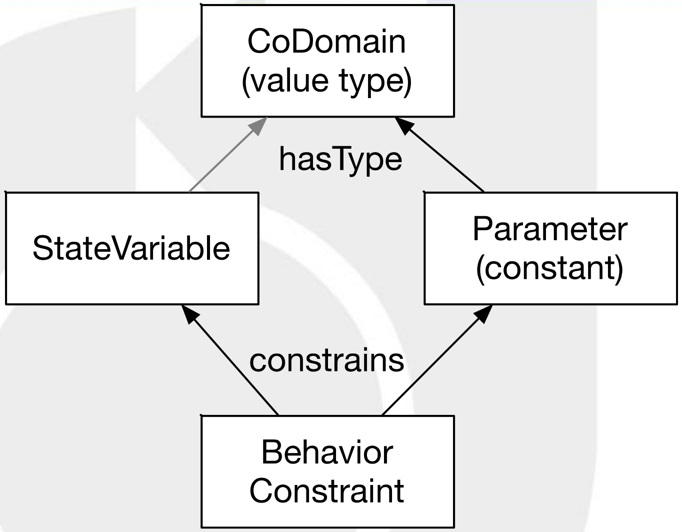

MagicDraw UML 12.5 - hibernate_sample.mdzip [C:\MD_no_install\MD125_0531\samples\AndroMDA\]
File Edit View Layout Diagrams Options Tools Analyze Teamwork Window Help
Composite-id
EJB Security
Hibernate Cartridge Test
Hibernate Collections
Hibernate Inheritance Te...
Containment
Properties
Ready
<<ApplicationException>>
TestException
<<Service>>
ServiceOne
+operationWithVoidReturnType() : void
+operationWithSimpleReturnType() : String
+operationWithComplexReturnType() : Collection
+SoperationWithSingleArgument(argumentOne : Date) : String
+operationWithMultipleArguments(firstArgument : Long, secondArgument : Boolean) : void
Tests generation of ju
<<Service>>
ServiceTwo
+attributeOne : String
+operationOne() : TestValueObject
<<ServiceElement>>+operationTwo() : String{@andromda.ejb.transaction.type = RequiresNew}
Tests generation of a
Test exception on entity operations.
<<Entity>>
EntityTwo
+attributeOne : Long = 12
<<Identifier>>+name : String
+operationOne(paramOne : String) : long
+operationTwo(paramOne : Long) : void
1..*
0..*
Test default identifier
Test required and non required association ends.
<<Entity>>
EntityOne
<<PersistentProperty>>+attributeOne : String{@andromda.persistence.column.index = "INDEX_1"}
+attributeTwo : int
+findByAttributeOne(attributeOne : String) : Collection{query}
<<FinderMethod>>+findByAttributeTwo(attributeTwo : int) : Collection{query,@andromda.hibernate.query = "from org.andromda.cartridges.hibernate.EntityOneas ent
+findByCollectionOfAttributeTwo(attributeTwos : Collection) : Collection{query}
Tests automatic finder generation
Tests many-to-many
<<Entity>>
EntityFour
{@andromda.hibernate.entity.dynamicInsert}
+enumTest : EnumerationOne
0..*
0..*
<<Entity>>
EntityFive
{@andromda.hibernate.version = "v

Ontological Approach

- Ontology establishes a vocabulary that can be used to talk about domain of interest
  - Focuses on the concepts and relationships of interest, not on the syntax or particular set of operators
- JPL’s institutional model-based systems engineering capabilities are being built upon an ontological modeling foundation
- BeCoS only presents information necessary for ontology

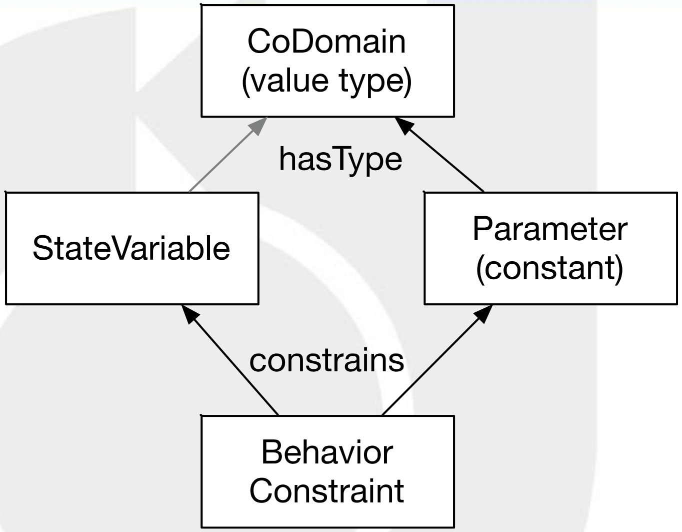
Ontology - Behavior

- **State variable** is a typed property whose value can change over time
- **Parameter** is a typed constant
- **Behavior constraints** constrain how state variable values can propagate over time
  - Constraints assert relationships that are true for all time

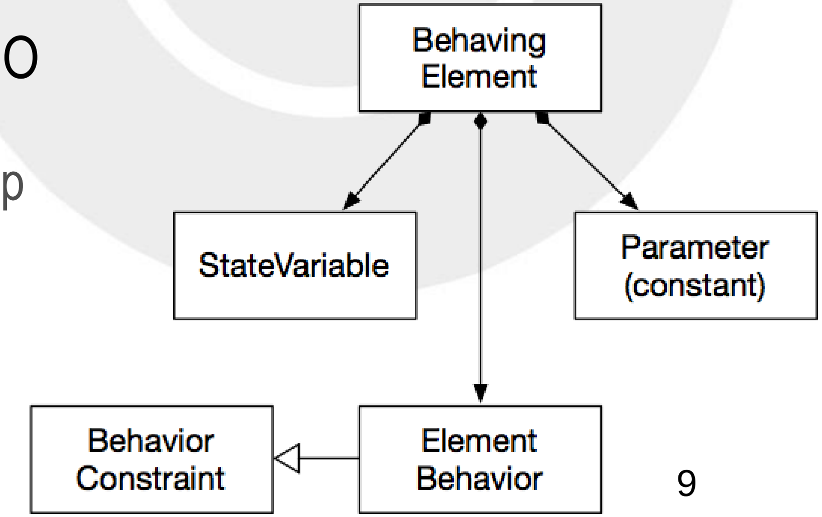


Ontology - Behavior

- **State variable** is a typed property whose value can change over time
- **Parameter** is a typed constant
- **Behavior constraints** constrain how state variable values can propagate over time
  - Constraints assert relationships that are true for all time

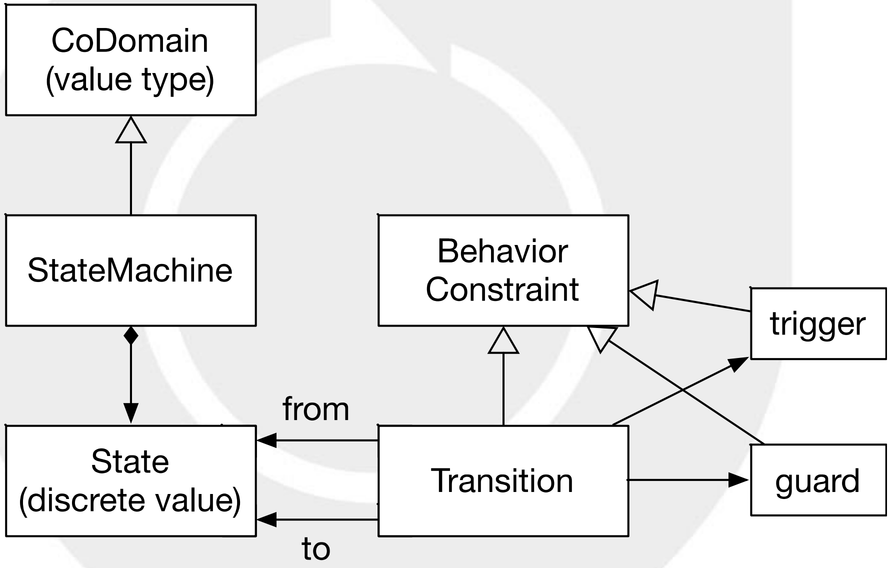


- **Behaving elements** simply provide an OO type containment of properties
  - Containment has no semantic relationship to the "math" of behavior, and is only intended to support an OO composition
  - Element Behavior is a form of Behavior Constraint (more specifically, it can be a set/container of Behavior Constraints)



Ontology - Behavior

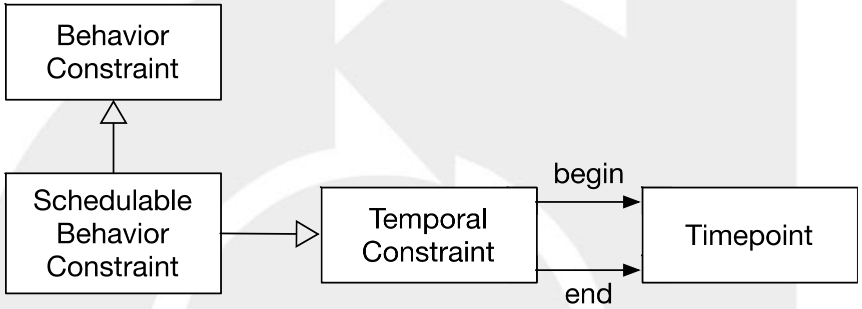
- State machines are a discrete value type
  - Useful abstraction for describing control behaviors
- SM defines a value type having an enumeration of orthogonal, discrete “state” values
- Transition rules define associated behavior constraints
  - Triggers, guards

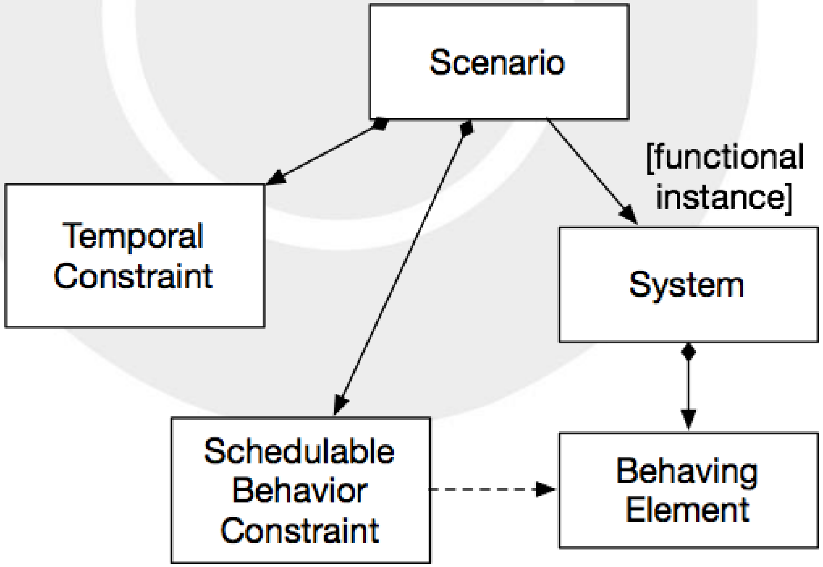


Ontology - Scenario

- A **scenario** describes some progression of states of a *particular system* over some unit of time

- A **scenario** is composed of:
  - schedulable behavior constraints on the states/parameters of the Behaving Elements in the System
    - E.g., “SwitchPosition = Closed” for 10 seconds
  - additional temporal constraints that serve to coordinate the behavior in time
    - E.g., “SwitchPosition = Open” for 5 seconds ***immediately precedes*** “SwitchPosition = Closed” for 10 seconds



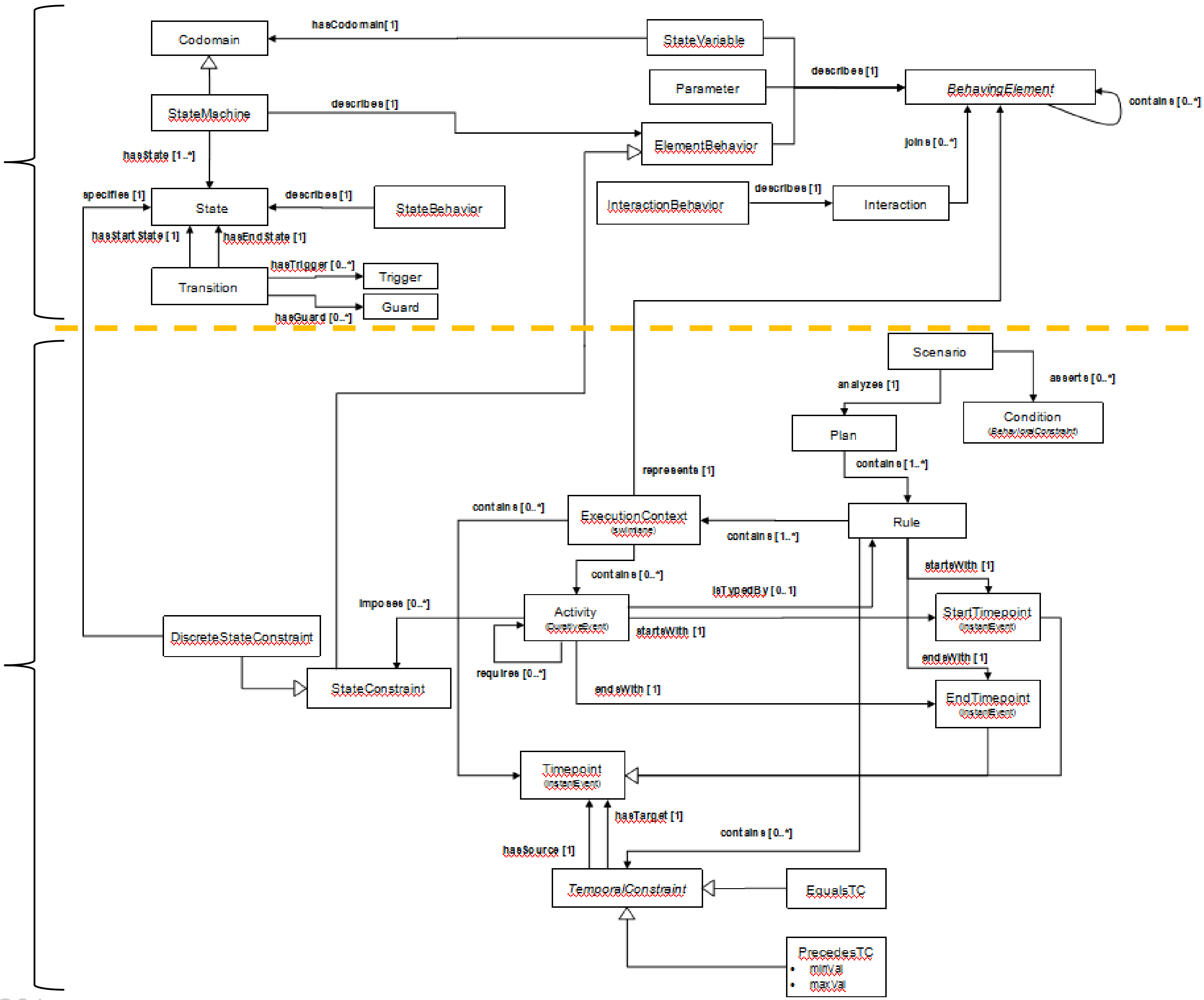


Shaping the Future of Aerospace

Ontology – Implemented in BeCoS

Behavior Ontology

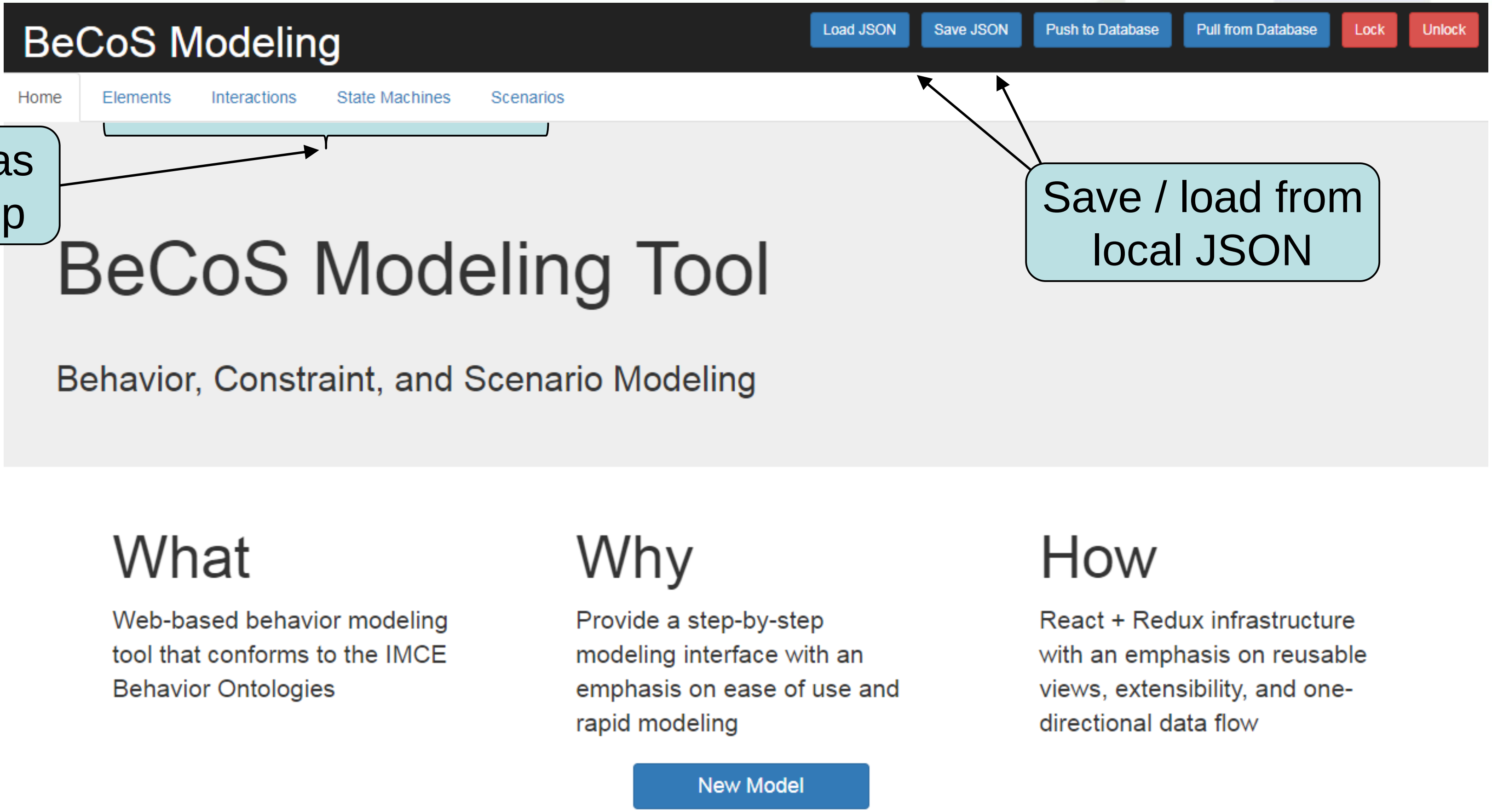
Scenario Ontology



BeCoS Tool

- Web-based application to support behavior modeling
  - Intrinsic behavior (behaving elements, state variables, parameters, state machines, constraints, interactions)
  - Scenarios (asserting how state constraints evolve over time)
- Architecture
  - React framework used with Redux
  - Off-the-shelf libraries used as much as possible
    - D3, Bootstrap, Mathquill, among others
- Guiding Principles
  - Site must be easily navigable and user-friendly
  - Present only relevant information to the user
  - Correct by construction enforced throughout app
    - Presenting only relevant information
    - Validation checks where applicable

BeCoS Tool



Four areas of the app

Save / load from local JSON

BeCoS Modeling Tool

Behavior, Constraint, and Scenario Modeling

What

Web-based behavior modeling tool that conforms to the IMCE Behavior Ontologies

Why

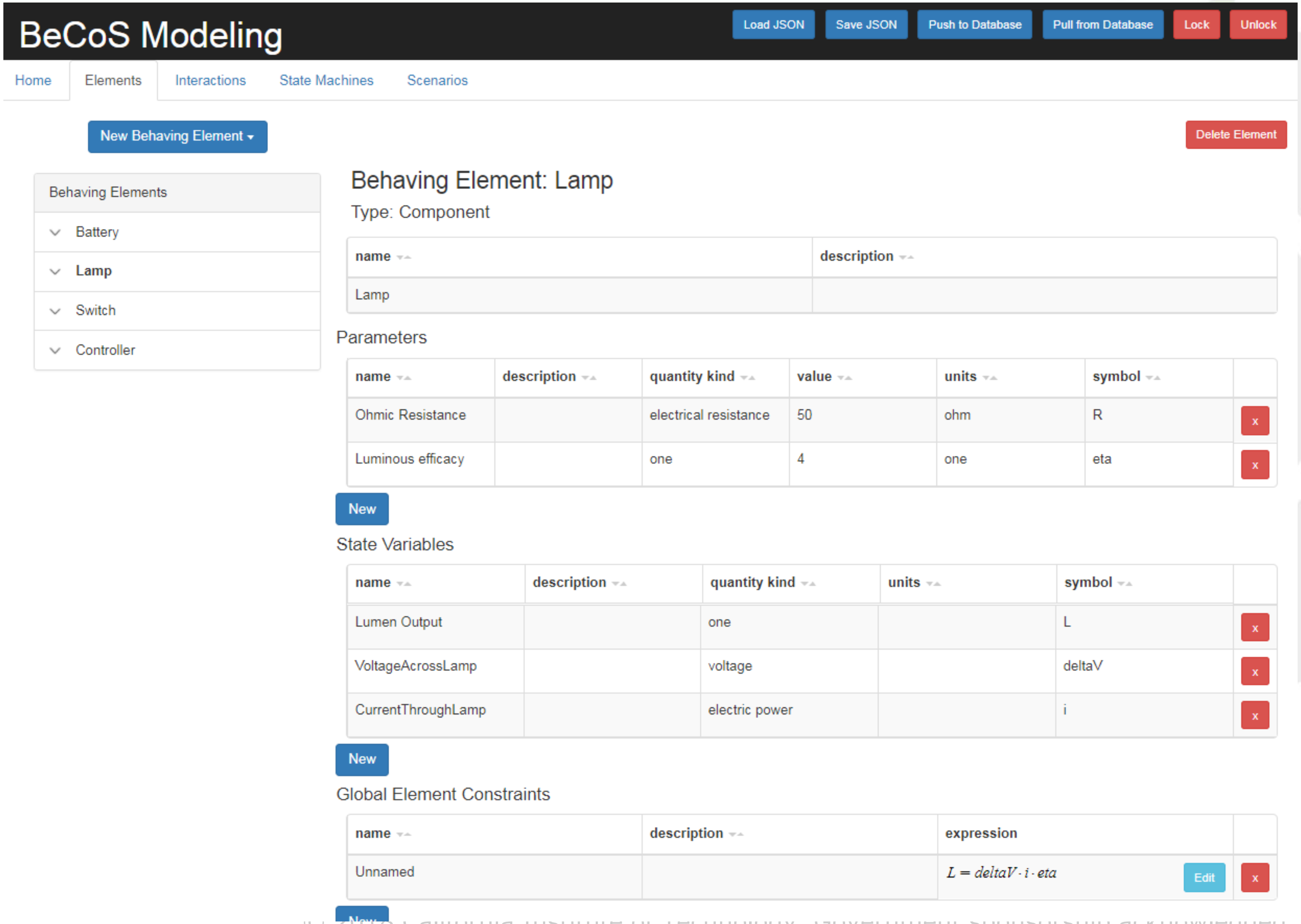
Provide a step-by-step modeling interface with an emphasis on ease of use and rapid modeling

How

React + Redux infrastructure with an emphasis on reusable views, extensibility, and one-directional data flow

- Illustrative example
  - Lamp circuit – battery, lamp, switch, controller

BeCoS Tool – Elements Tab



Behaving Element: Lamp

Type: Component

name	description
Lamp	

Parameters

name	description	quantity kind	value	units	symbol	
Ohmic Resistance		electrical resistance	50	ohm	R	x
Luminous efficacy		one	4	one	eta	x

State Variables

name	description	quantity kind	units	symbol	
Lumen Output		one		L	x
VoltageAcrossLamp		voltage		deltaV	x
CurrentThroughLamp		electric power		i	x

Global Element Constraints

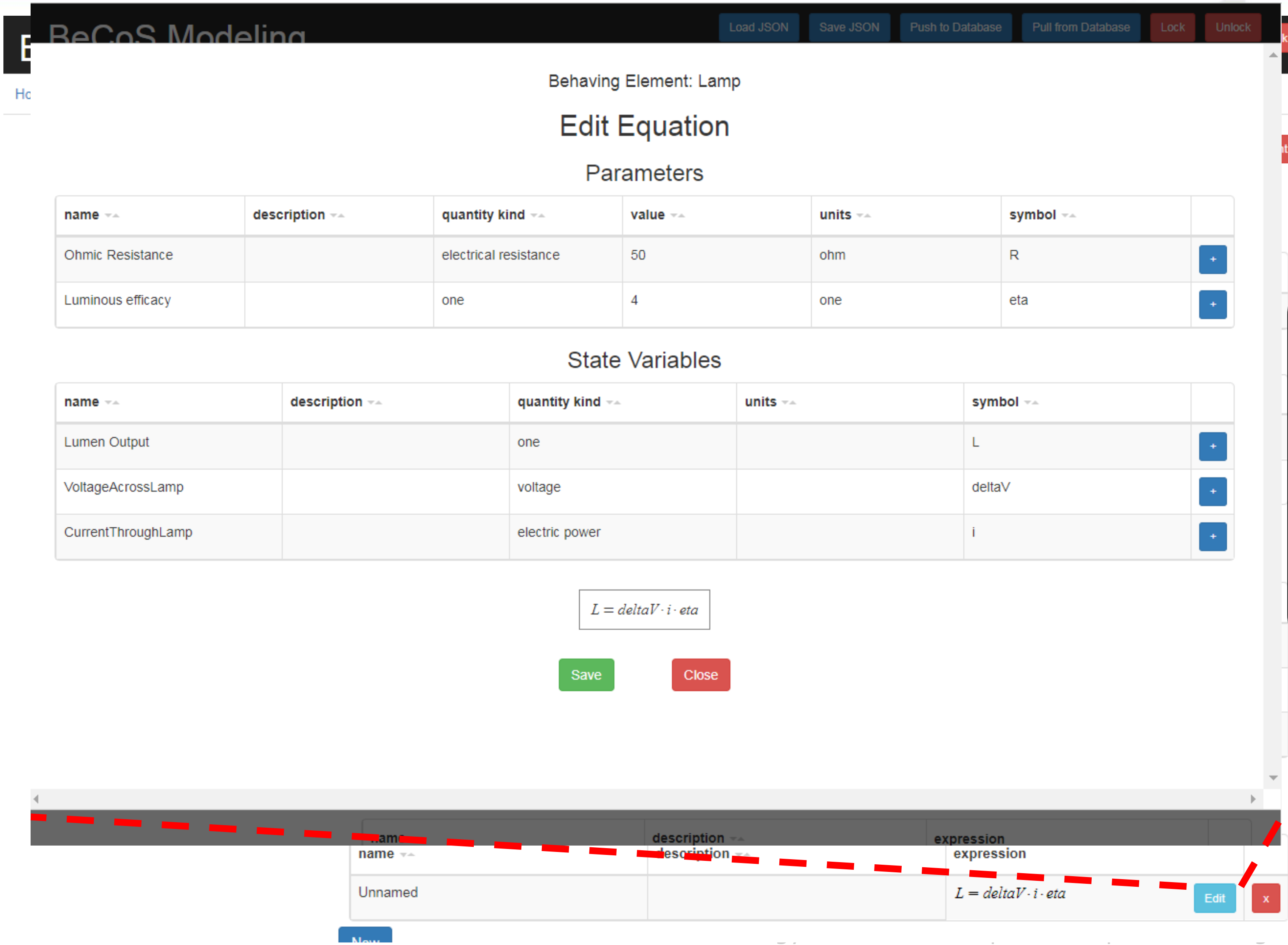
name	description	expression	
Unnamed		$L = deltaV \cdot i \cdot eta$	Edit x

Parameters (constants)

State Variables – time varying

Behavior constraint

BeCoS Tool – Constraint Editor



Behaving Element: Lamp

Edit Equation

Parameters

name	description	quantity kind	value	units	symbol	
Ohmic Resistance		electrical resistance	50	ohm	R	+
Luminous efficacy		one	4	one	eta	+

State Variables

name	description	quantity kind	units	symbol	
Lumen Output		one		L	+
VoltageAcrossLamp		voltage		deltaV	+
CurrentThroughLamp		electric power		i	+

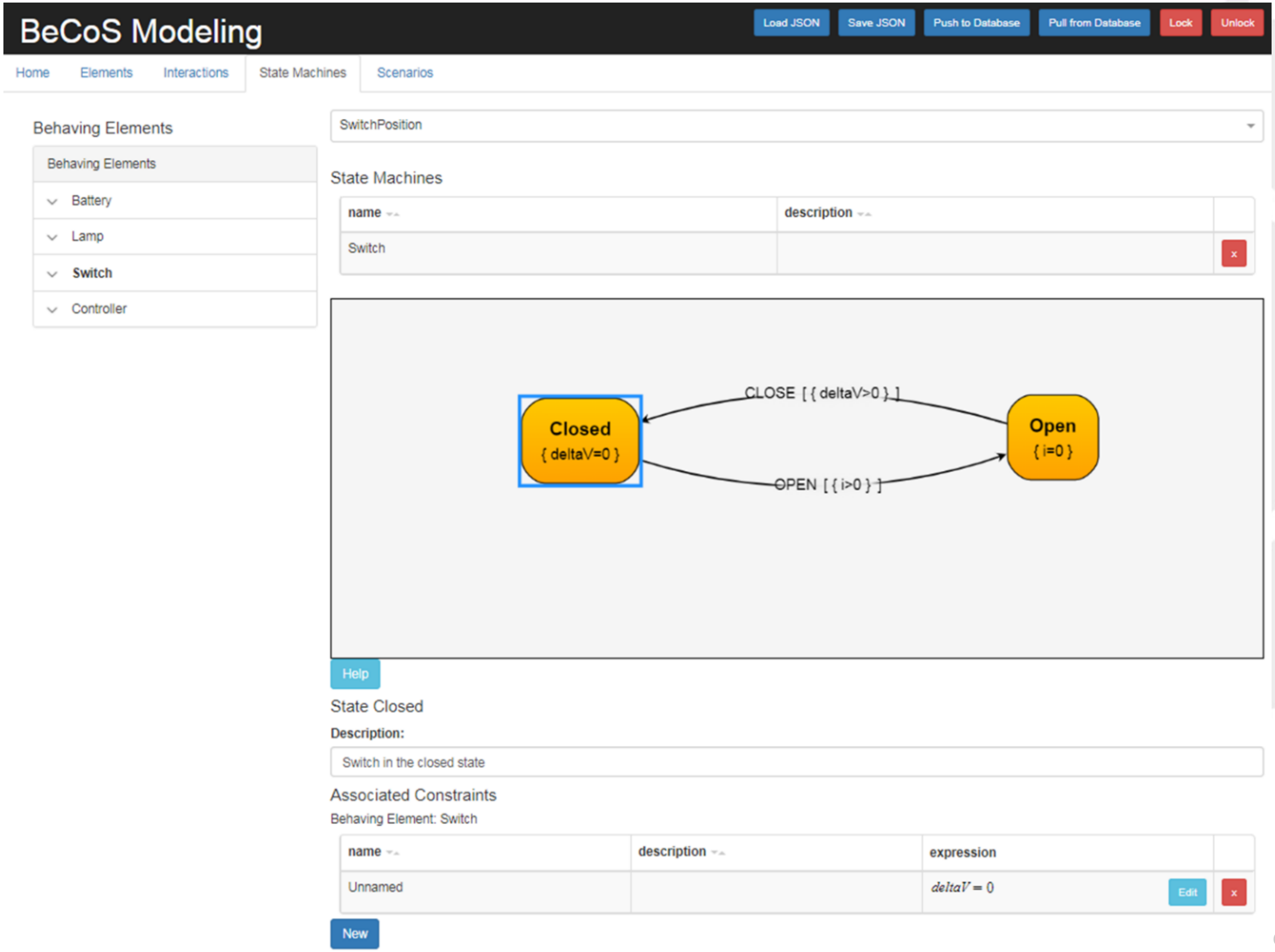
$$L = deltaV \cdot i \cdot eta$$

name	description	expression		
Unnamed		$L = deltaV \cdot i \cdot eta$	Edit	x

Defining behavior constraints –

Uses SVs and parameters from selected behaving element

BeCoS Tool – State Machines Tab



Behaving Elements

Behaving Elements
- Battery
- Lamp
- **Switch**
- Controller

SwitchPosition

State Machines

name	description
Switch	

State Closed

Description:

Switch in the closed state

Associated Constraints

Behaving Element: Switch

name	description	expression
Unnamed		$deltaV = 0$

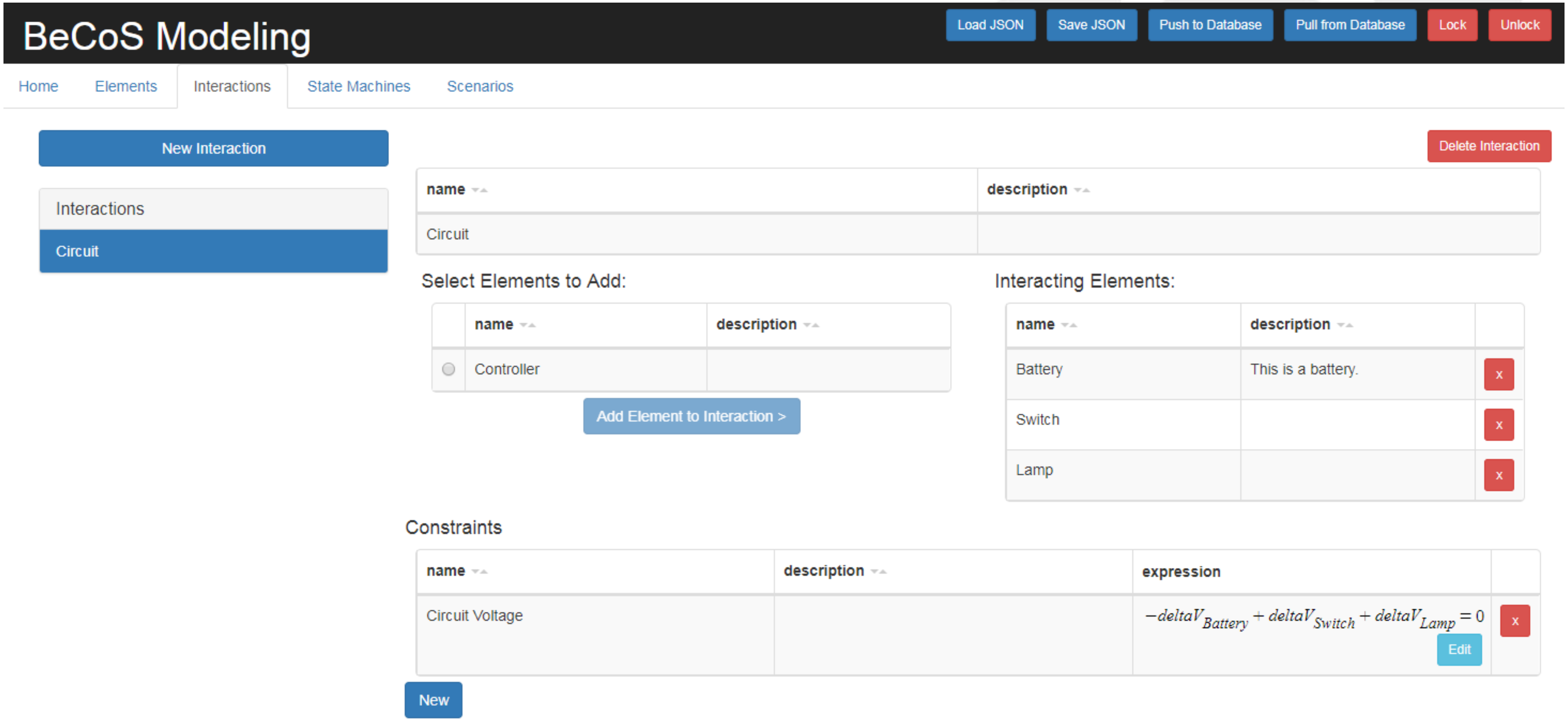
Switch SV “SwitchPosition” typed by “StateMachine”

- States with behavior constraints
- Transitions with triggers and guards

Defining behavior constraints on selected state (or trigger/guards on selected transition)

ed.

BeCoS Tool – Interactions Tab



name	description
Circuit	

Select Elements to Add:

	name	description
○	Controller	

Interacting Elements:

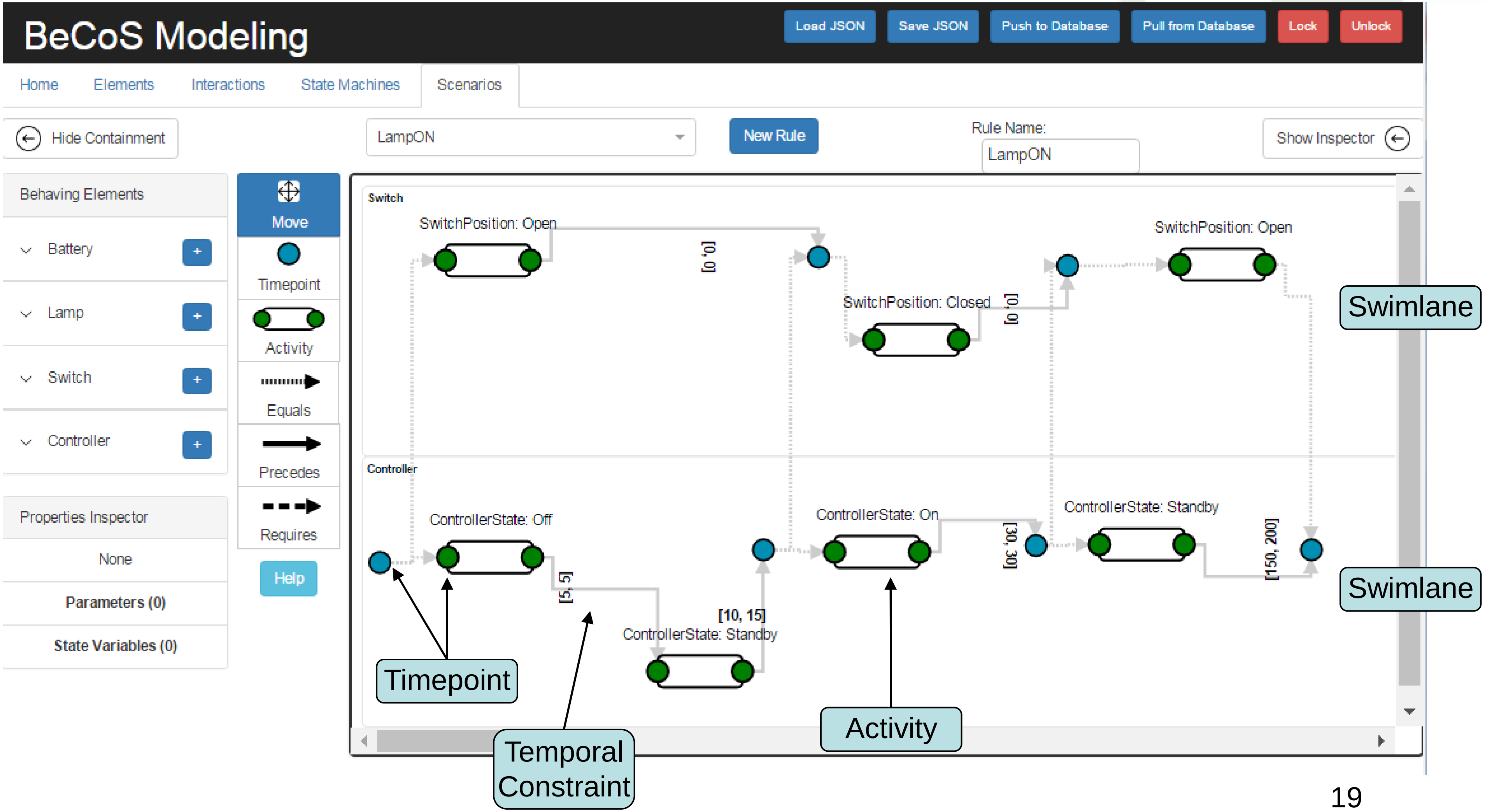
name	description	
Battery	This is a battery.	x
Switch		x
Lamp		x

Constraints

name	description	expression	
Circuit Voltage		$-deltaV_{Battery} + deltaV_{Switch} + deltaV_{Lamp} = 0$	x

Interactions define behavior constraints from more than one behaving element.

BeCoS Tool – Scenarios Tab



LampON

Rule Name: LampON

Swimlane

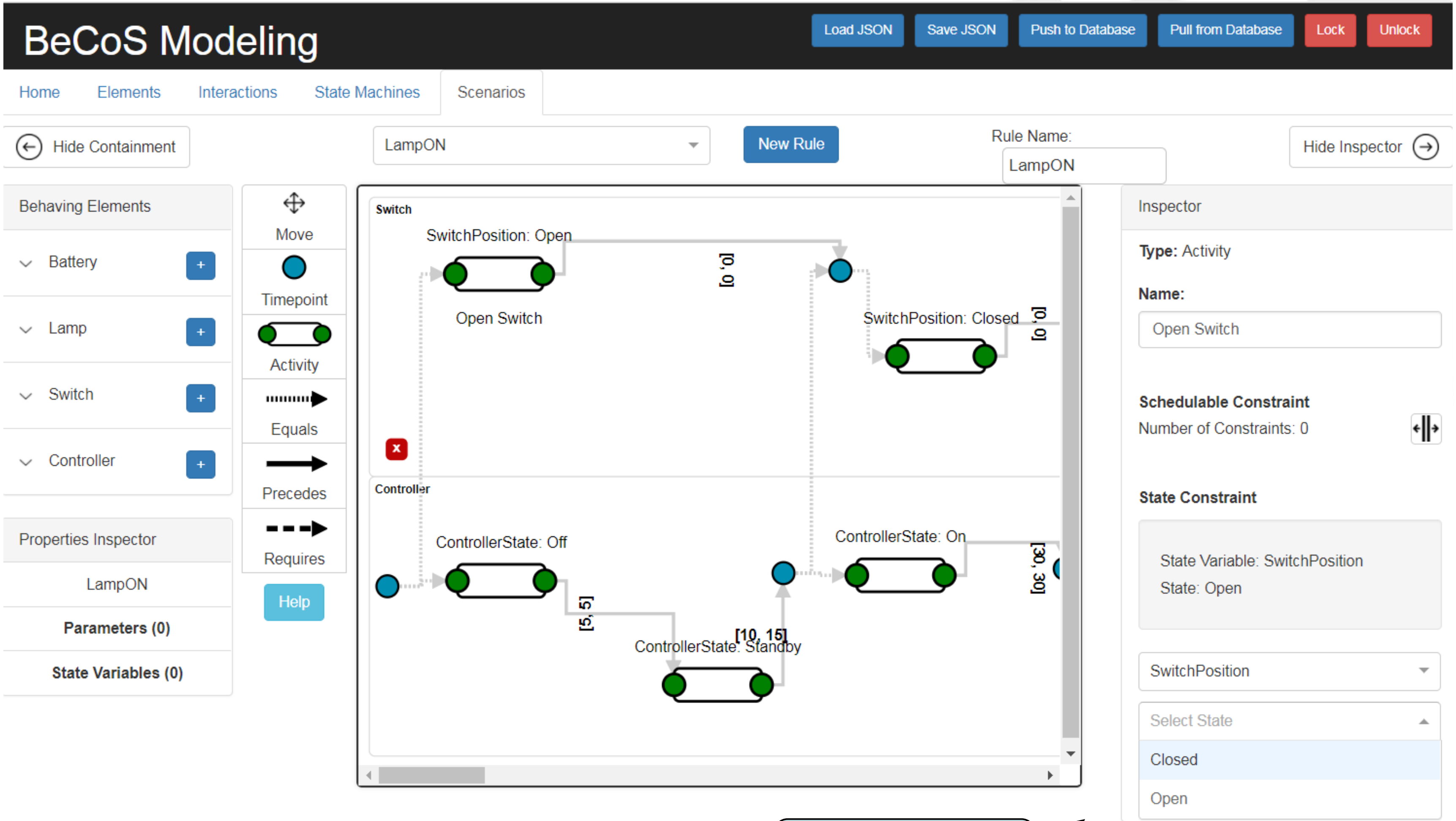
Swimlane

Timepoint

Temporal Constraint

Activity

BeCoS Tool – Scenarios Tab



LampON

Rule Name: LampON

Properties Inspector

LampON

Parameters (0)

State Variables (0)

Type: Activity

Name: Open Switch

Schedulable Constraint

Number of Constraints: 0

State Constraint

State Variable: SwitchPosition
State: Open

SwitchPosition

Select State

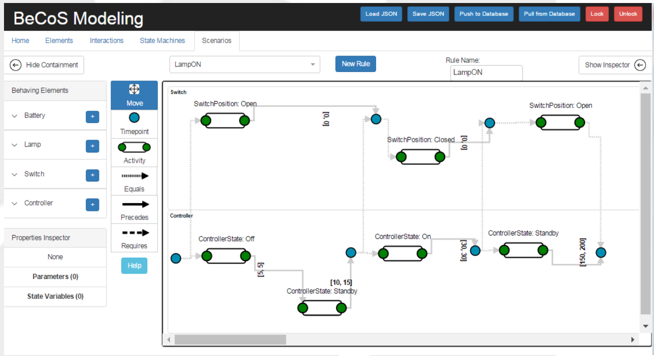
Closed

Open

Specify details of selected activity

Future Work

- Advancing tool to production quality
  - Fixing many identified issues (>75)
  - Assess user interface in Scenario tab



LampON

Rule Name: LampON

- Model validation analyses
  - State reachability analysis
    - Verifies system can transition into and out of all states
  - Scenario validation
    - ensure acyclic graph

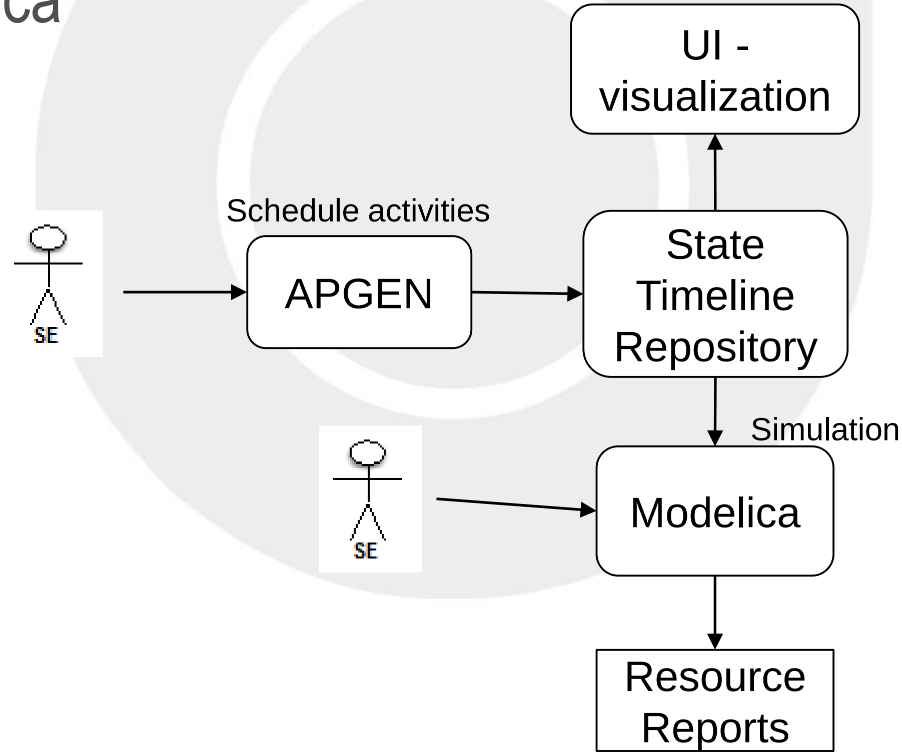
Future Work

- Integrate BeCoS into end-to-end resource analysis workflow

- Activity Plan Generator (APGEN)
  - Schedules activities, simulates a scenario, and produces state timelines for all modeled variables
  - Currently used as scheduling tool on Europa Clipper
  - Inputs to APGEN (system behavior and scenarios) must be hand-coded in a textual interface
    - Not possible to review inputs – only verification of modeled behavior is through inspection of timeline outputs
- JPL's new MBSE ecosystem
  - Tool / database to serve as single-source-of-truth
  - Exchange model information with specification tools (e.g. BeCoS, MagicDraw), and analysis tools (e.g. APGEN, Modelica)
    - Data exchanged adheres to ontologies

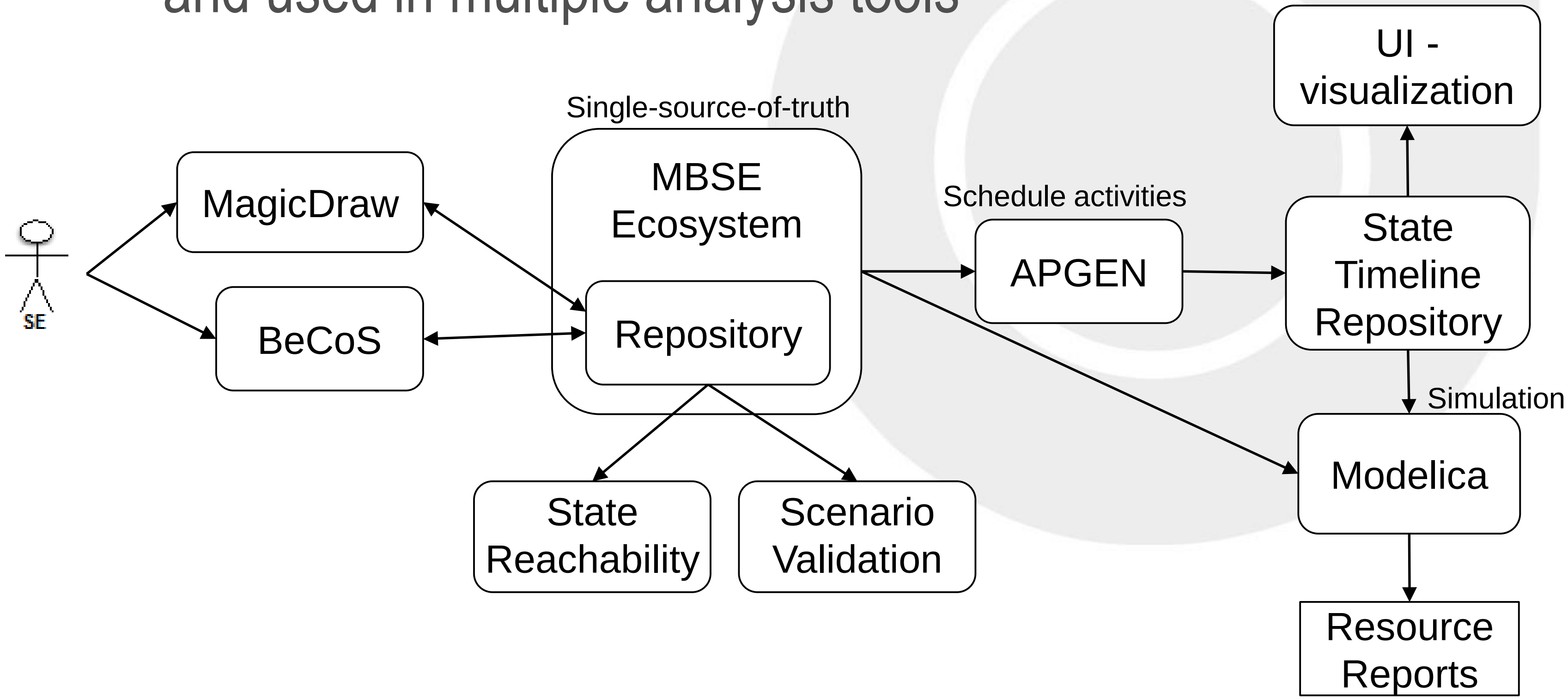
Integrating BeCoS with APGEN

- Current workflow for resource reports
  - Manual specification of behavior models in both APGEN and Modelica



Integrating BeCoS with APGEN

- Desired workflow for FY18
  - Behavior specified once in BeCoS can be transformed and used in multiple analysis tools



Summary

- Developed a web app to allow systems engineers to directly specify:
  - Behaviors (state variables, parameters, constraints, interactions, state machines)
  - Declaratively-specified scenarios
- Prototype tool with initial user testing by Europa Clipper users
- Plan to integrate BeCoS with JPL analysis tools
  - Perform simulations with behavior directly specified by systems engineers

Acknowledgements

- Team Members:
  - Justin Kaderka (task lead)
  - John Arballo (developer)
  - Tyler Ryan (past developer)
  - Erika Hill, Deanna Heer, Zachery Miranda, David Tsui, Thomas Kwak, and Brandon Wang (past intern developers)
  - Matthew Rozek (past user / advisor)
  - David Wagner (advisor)
  - Michel Ingham (advisor)

- Additional support provided by:
  - Steven Jenkins and Nicolas Rouquette (institutional IMCE ontology usage)
  - Jean-Francois Castet (behavior ontology, advised initial effort of the tool)

- **This work was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration**.

AIAA

Shaping the Future of Aerospace